A Path Metric Based Construction of Polarization-Adjusted Convolutional Codes

Tyler Kann[1], Shrinivas Kudekar, Matthieu Bloch[1]
[1] School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA
tkann3@gatech.edu, kudekar@gmail.com, matthieu.bloch@ece.gatech.edu

***Abstract*—We propose an approach to understand and exploit Polarization-Adjusted Convolutional (PAC) Codes that is directly tied to decoders with memory, specifically Successive Cancellation List (SCL) decoding. The crux of our approach is to use a modified Density Evolution Gaussian Approximation (DEGA) to account for errors in the decoding path and more accurately track the path metrics (PMs) likely to incur decoding errors. Our approach not only explains the benefits provided by the use of the rate one precoding, but also provides new insight into why certain information sets perform better under PAC than polar. We leverage the approach to design new information sets, and in particular, we design a $(128, 42, L = 8)$ code that offers half a dB gain over the state-of-the-art at a Frame Error Rate (FER) of 10^{-3} and outperforms the Reed-Muller (RM) set with $L = 32$. We also draw connections between our approach and works studying the minimum weight of PAC codes.**

I. Introduction

As engineering pushes towards the realization of 5G New Radio and 6G, many challenges arise. One of these is the need for communications with improved reliability and less latency, notably the need for the Ultra-Reliable Low-Latency Communication (URLLC) requirement. These strict requirements demand new and improved error-control coding for extremely short blocklengths, typically $N \leqslant 256$ [1]. Promising coding schemes that have shown strong performance include extended Bose-Chaudhuri-Hocquengham (eBCH) [2]; RM codes [3]; polar codes [4] and Polarization-Adjusted Convolutional (PAC) codes [5]–[7], especially when used in combination with decoding techniques such as Successive Cancellation List (SCL) decoding [7], [8]; Ordered-Statistics Decoding (OSD) [9]–[11]; or Guessing Random Additive Noise Decoding (GRAND) [12].

While several papers have exploited the benefits of PAC codes for large improvements at short blocklengths, there have been few attempts to develop a principled explanation for the boost in performance PAC provides over traditional polar coding. Existing attempts have focused on cutoff metrics [13], a joint source-coding perspective [14], and primarily on the impact the precoder has on the weight spectrum of the code [15], [16]. However, none of the current methods in the literature explain why the benefit of PAC is tied directly to the decoder, or more explicitly, why PAC benefits are not realizable under Successive Cancellation Decoding (SCD) but are attained under SCL, Fano, or any other decoder that tracks multiple paths. In this work, we attempt to explain PAC in a way that is explicitly tied to how the aforementioned decoders track PMs. Understanding how PAC codes change the PM may explain why they perform so well for these decoders and may also pave the way to create improved information sets and convolutional precoders.

The remainder of the paper is organized as follows. We review necessary notation and concepts regarding polar codes, DEGA, and PAC in Section II. We then introduce conditional DEGA, our main tool to understand the behavior of PAC codes in Section III, and our proposed PM based information set construction in Section IV. A full algorithm and resulting frozen sets are available upon request, but for brevity Section IV merely highlights the salient features of our approach. We conclude by presenting numerical results in Section V.

II. Polarization-Adjusted Convolutional Codes

A. Polar Codes

A polar code for channel coding is characterized by its blocklength $N \triangleq 2^n, n \in \mathbb{N}$, the number of information bits K, and an information set $\mathcal{A} \subset [1; N]$ that specifies where to place information bits. Specifically, a vector of K information bits m is encoded into a length N vector u such that $u_{\mathcal{A}} \triangleq m$ and $u_{\mathcal{A}^c} \triangleq 0$, the all-zero vector. The sets $\mathcal{A}$ and $\mathcal{A}^c$ are called the information and frozen set, respectively. Upon setting $G = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$, the base polarization matrix, and $G^{\otimes n}$ as the nth order Kronecker product of G, a codeword x is created through the operation $x \triangleq uG^{\otimes n}$, a process known as the polar transform. The structure of the matrix $G^{\otimes n}$ allows for an encoding complexity of $O(N \log N)$. The codeword x is then sent over the channel to a receiver that observes a noisy version y, e.g., $y_i = x_i + n_i, n_i \sim \mathcal{N}(0, \sigma^2)$ if the channel is an Additive White Gaussian Noise (AWGN) channel.

The standard decoding algorithm for polar codes is the Successive Cancellation (SC) decoder, by which bits from the information set $\mathcal{A}$ are successively decoded based on their Log-Likelihood Ratio (LLR), λ, given the past decoded bits according to the maximum-likelihood rule.

$$\forall i \in \mathcal{A} \quad \hat{u}_i^0 = \begin{cases} 0 & \text{if } \lambda_i^0 = \ln \frac{P(y, \hat{u}_{0:i-1} | \hat{u}_i^0 = 0)}{P(y, \hat{u}_{0:i-1} | \hat{u}_i^0 = 1)} > 0 \\ 1 & \text{else.} \end{cases}, \quad (1)$$

$$\forall i \in \mathcal{A}^c \quad \hat{u}_i^0 = 0, \quad (2)$$

This works was supported in part by the National Science Foundation (NSF) under grant 2148400 as part of the Resilient & Intelligent NextG Systems (RINGS) program.

where $\hat{u}_{0:i-1} = \{\hat{u}_0^0, \hat{u}_1^0, \ldots, \hat{u}_{i-1}^0\}$ denotes the vector of past decisions. The LLRs can be efficiently computed recursively, resulting in a decoding complexity of $O(N \log N)$. The LLRs can also be viewed as decisions made at the output of individual *bit channels*, corresponding to channels with input bit u_i and output $(y, \hat{u}_{0:i-1})$. The choice of the set $\mathcal{A}$, called *rate profiling*, plays a crucial role in determining the performance of polar codes, and several criteria have been proposed based on the capacity of the bit channels [17] or the RM profile [18], which consists in enumerating the Hamming weight of RM codewords and selecting those with highest weight, essentially giving rise to the Reed-Muller codes.

B. Successive Cancellation List Decoding

Given that each decision in the SC decoder relies on previously decoded bits, one bad decision can propagate and make future bits erroneous. SCL decoding [19] attempts to avoid this problem and can be implemented relatively efficiently. In SCL, the decoding process is viewed as following branches of a tree, simultaneously tracking up to L branches. The tree splits at every non-frozen bit i, creating two branches: $\hat{u}_i = 0$ and $\hat{u}_i = 1$. For every path in the set $\mathcal{L}$ of currently tracked paths, the PM is computed as the accumulation of Branch Metrics (BMs). Each BM is the penalty based on the decision for both frozen and information bits, according to:

$$\forall l \in \mathcal{L}, \quad \text{PM}_i[l] = \text{PM}_{i-1}[l] + \text{BM}_i[l] \tag{3}$$

$$\text{BM}_i[l] = \begin{cases} 0 & \text{if } 1 - 2\hat{u}_i^0[l] = \text{sign}(\lambda_i^0[l]) \\ |\lambda_i^0[l]| & \text{if } 1 - 2\hat{u}_i^0[l] = -\text{sign}(\lambda_i^0[l]) \end{cases}. \tag{4}$$

When $|\mathcal{L}|$ is greater than L, the list is pruned back to only L paths, keeping those with the lowest path metrics.

In list decoding, there are two possible error events that result in a decoding failure. The first error event occurs when the paths are pruned from $2L$ to L, and the correct path is deleted in this process. The second error occurs at the end, when the correct path is one of the L remaining, but not the path with the lowest PM. We refer to these errors as path loss errors and selection errors, respectively. We are primarily considering the selection error. The selection error occurs when there is at least one path with a smaller PM than the correct path. With a slight abuse in terminology, we refer to both path and metric of the incorrect path with the smallest PM as the Most Misleading Path Metric (MMPM). This is the path that determines if there is a selection error, which happens if and only if the MMPM is less than the correct PM. Thus our guiding principle is to increase the value of the MMPM and consequently reduce the number of selection errors.

C. Density Evolution Gaussian Approximation

Ultimately, one of the hardest challenges of both polar and PAC codes is creating the information set $\mathcal{A}$ that minimizes the block error probability.

This boils down to choosing information bits that are deemed the most reliable according to the chosen metric of channel reliability. One method of estimating the reliabilities of polarized channels is using DEGA, based on [20] and applied to polar codes in [21], [22]. In essence, the received LLRs are treated as Gaussian distributions, parameterized as $\mathcal{N}(\frac{2}{\sigma_N^2}, \frac{4}{\sigma_N^2})$, where σ_N^2 is the noise variance of the channel. We aim to calculate the *posterior* distributions at the output. With an abuse of notation, we also denote L as the distribution of the LLRs, and reserve λ for the realizations. Since the distribution is symmetric, meaning the variance is twice the mean, only one parameter must be accounted for. The distributions then go through the polarization, and we calculate:[1]

$$L_N^{(2i-1)}(y_1^N, \hat{u}_1^{2i-2}) = \tanh(L_{N/2}^{(i)}(y_{N/2+1}^N, \hat{u}_{1,e}^{2i-2})/2)) \times 2\tanh^{-1}(\tanh(L_{N/2}^{(i)}(y_1^{N/2}, \hat{u}_{1,e}^{2i-2} \oplus \hat{u}_{1,o}^{2i-2})/2) \tag{5}$$

$$L_N^{(2i)}(y_1^N, \hat{u}_1^{2i-1}) = L_{N/2}^{(i)}(y_{N/2+1}^N, \hat{u}_{1,e}^{2i-2}) + (-1)^{\hat{u}^{2i-1}} L_{N/2}^{(i)}(y_1^{N/2}, \hat{u}_{1,e}^{2i-2} \oplus \hat{u}_{1,o}^{2i-2}) \tag{6}$$

Given that we are transmitting and correctly decoding the all zero codeword, *all* repetition nodes, i.e. (6), correspond to addition of LLRs. Compared to complex Low Density Parity Check codes, polar codes form, by design, a perfect tree with no cycles and exactly two inputs in the density evolution. This makes the posteriors not only easier to calculate but also accurate, which is used to predict the decoding performance.

D. PAC Codes

While polar codes are asymptotically optimal [4], they are not immediately competitive in the short to medium length regime. This is because the polarization process, by which channels eventually become either completely noisy or noiseless, happens relatively slowly, and smaller codes contain channels that do not yet fall in either of these categories. To improve the performance in the short to medium length regime, Arıkan introduced Polarization-Adjusted Convolutional (PAC) codes in [5], which can essentially be viewed as polar codes with dynamically frozen bits [22], [23]. The surprising revelation of PAC codes is that precoding the polar code with an outer *rate one* convolutional code offers dramatic improvements in performance with no rate penalty.

PAC codes still ultimately polarize u to get the final codeword, except u is the now output of the convolution of v, the new information vector. The information set $\mathcal{A}$ now determines the placement of information bits, so that $v_{\mathcal{A}}$ contains the information bits and $v_{\mathcal{A}^c} = 0$. A detailed explanation of PAC codes can be found in [24]. Decoding operates on $\hat{u}_i$, except $\hat{u}_i$ is now determined from the convolved $\hat{v}_{0:i}$.

III. Conditional DEGA

DEGA, as described in Section II-C, considers the all-zero codeword being transmitted, and the reliability of each bit is based on the assumption that all prior bits have been decoded correctly. This assumption is sufficient for SCD, in which one cannot afford to make any error. However, the

[1]We keep the DEGA density equations unmodified from [21], meaning the subscript and superscript are different from than the rest of the paper.

improved performance of decoders that track multiple paths (Fano, Stack, SCL, etc.) motivates us to investigate what happens to posteriors conditioned on an error made earlier. If a decoding error is made, an erroneous 1 now exists in the all-zero codeword.

Let us see how this erroneous 1 affects the propagation of LLRs when computing the posterior LLR for a future bit. Note that we have degree two repetition nodes and parity-check nodes in the decoding tree. As a result of polarization, many future repetition nodes (at all layers) will now be updated as

$$L_N^{(2i)}(y_1^N, \hat{u}_1^{2i-1}) = L_{N/2}^{(i)}(y_{N/2+1}^N, \hat{u}_{1,e}^{2i-2}) - L_{N/2}^{(i)}(y_1^{N/2}, \hat{u}_{1,e}^{2i-2} \oplus \hat{u}_{1,o}^{2i-2}). \quad (7)$$

We are now *subtracting* the two positive mean inputs. Up until this point, because of the structure of the problem, both $L_{N/2}^{(i)}(y_{N/2+1}^N, \hat{u}_{1,e}^{2i-2})$ and $L_{N/2}^{(i)}(y_1^{N/2}, \hat{u}_{1,e}^{2i-2} \oplus \hat{u}_{1,o}^{2i-2})$ are identical densities. This means that the new distribution resulting from (7) will be of the form $\mathcal{N}(0, 4\mu)$, where 2μ is the variance of the distribution of $L_{N/2}^{(i)}(y_1^{N/2}, \hat{u}_{1,e}^{2i-2} \oplus \hat{u}_{1,o}^{2i-2})$. Because of the tree structure of repetition nodes and check nodes in polar codes, the expectations of all future LLRs shrinks after the first error, with many becoming zero. An example of this occurrence is illustrated in Fig 1. We acknowledge that [25] identified a similar phenomenon, but did not leverage it or explicitly show what happened to future LLRs.

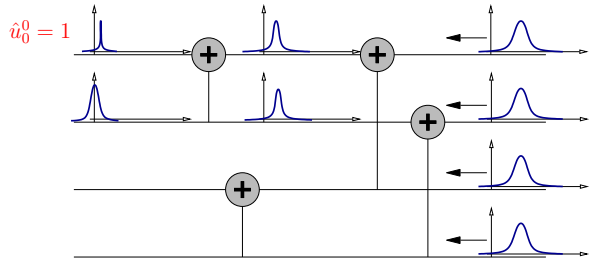
Calculating the propagation of conditional DEGA on a check node becomes harder than traditional DEGA because the distributions are no longer symmetric and one needs to account for a variance that is now vastly different from the smaller mean. In this work, we calculate the output mean and variance through check nodes empirically via monte carlo. Since the distributions are no longer symmetric, posteriors may have an extremely small, if not zero, mean but large variance. Even though the distributions are no longer symmetric, we assume that all inputs and outputs of the check nodes and repetition nodes are Gaussian. If the conditional posterior of bit i comes from a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$, the expected BM can then be calculated with one of the two integrals

$$\mathbf{E}[\text{BM}_i] = \begin{cases} \int_0^{+\infty} |\lambda| \mathcal{N}(\mu, \sigma^2) \\ \int_{-\infty}^0 |\lambda| \mathcal{N}(\mu, \sigma^2) \end{cases}, \quad (8)$$

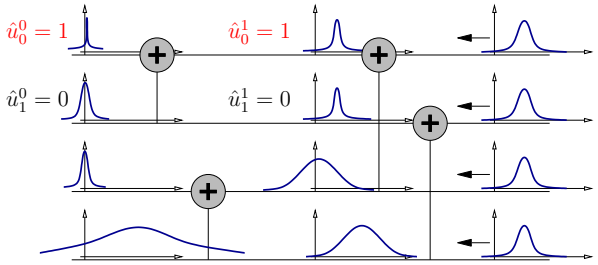
depending on the the optimal decision for u (i.e. the sign of μ). Note that for frozen bits, only the top case is allowed. Since the PM is simply a sum of all branch metrics, it can be estimated as

$$\mathbf{E}[\text{PM}] = \sum_{i=0}^{N-1} \mathbf{E}[\text{BM}_i]. \quad (9)$$

For information bits, since the decoder makes decisions based on the posterior LLR and BM, the greedy choice causes no notable increase in the PM. This means that majority of the $\mathbf{E}[\text{PM}]$ comes from the frozen bits with a large $\mathbf{E}[\text{BM}]$, which are ones with a small conditional mean but large variance, or ones with a large conditional mean and a forced decision of 1 via PAC.



(a) The received LLRs are on the "right" and the posterior LLRs are computed on the "left". In traditional DEGA, the distributions are symmetric and shrink towards zero when going through a check node and move towards $+\infty$ when going through a repetition node. Because of the distribution symmetry, as the distributions move towards zero, the variance also shrinks to zero. We assume here that the first bit is erroneously decoded to 1. This causes subtraction on the repetition node, and the resulting distribution of the posterior for bit 1 has zero mean.



(b) Because of the erroneous 1, the second layer polarizes to $(1, 0)$. This causes subtraction amongst the received LLRs. The distribution of the posterior on bit 2 is zero mean but not zero variance; the posterior of bit 3 is distributed by $\mathcal{N}(2\mu, 8\mu)$, since the inputs to the repetition node are $\mathcal{N}(2\mu, 4\mu)$ & $\mathcal{N}(0, 4\mu)$.

Fig. 1: Illustration of conditional DEGA for an error in $N = 4$.

As illustrated in Fig 1, in a scenario where all bits up to i have been decoded but an error was made on the ith bit, DEGA becomes inaccurate in estimating the posteriors of the future bits. Thus, computing the conditional DEGA distributions of future bits is essential for estimating the PM of this error path.

A. Relation between Conditional DEGA and Code Weight

As seen in Fig 1, some of the conditional posteriors now have zero mean. Greedy decisions on these bits do not alter the expected PM. Additionally, the decisions on such bits do not alter the magnitude of the future expected LLRs, although they may change the sign. In other words, neither decision is viewed as incorrect from the perspective of the decoder. These bits are problematic since they have equal probability to be 1 or 0. This means they can interfere with the precoding, either by creating even parity and not altering a frozen bit decision, or by swapping the sign of a frozen bit that will be altered by PAC. Both scenarios limit the potential increase in PM incurred by PAC and create paths that have less than expected PMs.

We compare the finding of zero mean posteriors to [15], in which Rowshan et al. define the set

$$\mathcal{K}_i \triangleq \{j \in [i+1, N-1] \colon \mathrm{w}(\mathbf{g}_j) \geqslant \mathrm{w}(\mathbf{g}_i + \mathbf{g}_j) = \mathrm{w}(\mathbf{g}_i)\}, \tag{10}$$

where $\mathrm{w}(.)$ is the Hamming weight, and $\mathbf{g}_j$ is row j in the $G^{\otimes n}$ matrix. In words, $\mathcal{K}_i$ is the subset of all row vectors of $G^{\otimes n}$ whose weight, when linearly combined with $\mathbf{g}_i$, does not affect the resulting Hamming weight. Rowshan et al. use these sets to help with min-weight codewords enumeration. One can also define the set

$$\mathcal{K}_i^* \triangleq \{j \in [i+1, N-1] \colon \mathrm{w}(\mathbf{g}_j) < \mathrm{w}(\mathbf{g}_i + \mathbf{g}_j) = \mathrm{w}(\mathbf{g}_i)\}. \tag{11}$$

Empirically, $\mathcal{K}_i \cup \mathcal{K}_i^*$ seem to correspond to all the bit positions that have a zero mean conditional posterior given an error on i. In other words, deciding $\hat{u}_j$ incorrectly once an error has been made on $\hat{u}_i$ affects neither the resulting PM *nor* codeword weight. This suggests strong connections between our work on conditional DEGA and [15]. This does not mean, however, that two codewords with the same weight have the same PM; the PM should therefore offer more information. Additionally, the PM is directly tied to list decoding, whereas other metrics, including weight are not. Yet, it can be shown that the benefits of PAC are not achieved with SCD, thus we believe that while similar, the PM is a more accurate metric to look at. Regardless, since weight appears to be a closely tied proxy to PM, we do believe the work in [15] could be used as a valuable tool to creating sets aligning with our design.

B. Relation of PAC and Conditional DEGA

The decoding of PAC codes can be viewed as normal decoding of polar codes except each bit u_i is now decided as a function involving v_i and some subset of $v_{0:i-1}$. The choice of subset of $v_{0:i-1}$ comes from the choice of the generator. In this way, PAC codes contain *dynamically frozen bits*. Thus, if there is an error in deciding an information bit, that error propagates through the generator. For well-designed information sets and generator connections, the PAC decoder alters the decision of some frozen bits with non-zero conditional posterior, thus increasing the PM of the incorrect path. Note that while one is able to alter the decision of information bits, given that we will always have the choice of both $\{0, 1\}$, the SCL decoder merely favors the choice whose branch metric did not increase. Therefore, we believe that ultimately this ability of PAC being to alter the decision of frozen bits is the underlying reason of its success, and additionally why it is dependent on decoders that can track multiple paths.

IV. Creation of a PM Based Information Set

We now propose an algorithm to create information sets more suitable to exploit the benefits of PAC and SCL using conditional DEGA as our primary tool. As previously stated, DEGA works best for SCD, since one operates under the assumption that everything has been decoded correctly, and no mistakes are allowed. Since multiple mistakes are allowed under SCL, we propose a set that does not have the largest DEGA score, but rather has the largest potential MMPM under PAC. To do this, we compute the PM caused by an error made on bit i (not just the BM as with DEGA), which involves all the conditional DEGA based BMs as in (9) along with additional penalties that can be forced on by the convolutional precoder. Our goal is to find the information set $\mathcal{A}$ with the largest MMPM. Thus our optimization of the information set is a max min problem, specifically:

$$\underset{\mathcal{A} \in \binom{N}{K}}{\operatorname{argmax}} (\mathrm{MMPM}), \mathrm{MMPM} \triangleq \min_{i \in \mathcal{A}} (\mathrm{PM}_i) \tag{12}$$

where PM_i is the PM from an error on an information bit $i \in \mathcal{A}$ and other additional penalties incurred from PAC. This is a hard combinatorial optimization problem and we introduce several heuristics to find a good information set, chief amongst them being the use of conditonal DEGA to compute the PM. Note that in Fig 1, an error was made on bit 1, and now bits 2 and 3 have zero posterior. Incorrectly decoding bits 2 or 3 does not alter the PM of this error path. However, it may be the case that making an error on *just* 2 or 3 results in a larger PM than errors on both $\{1, 2\}$ or $\{2, 3\}$, because these bits may have a larger unconditional DEGA mean, meaning the resulting PM may be higher. This indicates that the largest PM possible from PAC is not obtained by forcing an error on all frozen bits with non-zero conditional posterior. This means that finding the set that maximizes the PM would require searching over $2^{K_i'}$ possibilities, where K_i' is the number of conditionally non-zero frozen bits given an error on i. To drastically simplify the search space, we only look at the frozen bit with the largest conditional mean LLR, reducing the search space down to one option. Additionally, we also occasionally look at limiting the search of the best frozen bit to only frozen bits between $[i, j]$, where j is m (a design parameter) information bits in the future, in an attempt to adhere more strictly to the list size constraint.

The following is a brief explanation of how we search for good information sets. In essence we attempt to find a local optimum of the optimization problem. We first make our set $\mathcal{A}$ of size K based on traditional DEGA. We also calculate the next ϵ best bits to store as our $\mathcal{F}' \subseteq \mathcal{F}$, which are bits currently not in $\mathcal{A}$ but are viable candidates. The goal is now to find suitable replacements for the bits in the $\mathcal{A}$ from $\mathcal{F}'$. A larger ϵ searches more of the $N - K$ remaining bits, but causes longer run time. A large number of the $N - K$ have small posteriors, making them unviable and thus do not need to be considered, so a small ϵ is sufficient. We then calculate $\mathrm{PM}_i \ \forall i \in \mathcal{A}$ using the description above with a desired m. The bit with the lowest PM is removed from $\mathcal{A}$ and added to $\mathcal{F}'$, as it has the worst PM, meaning it is highly likely to cause an error under SCL PAC. We then calculate $\mathrm{PM}_j \ \forall j \in \mathcal{F}'$, in a process exactly the same as the information bits. The *largest* of these we deem as the best suitable replacement, and move the index from $\mathcal{F}'$ to $\mathcal{A}$. Because sets lose potentially important frozen bits, this change does not guarantee an increase in the

minimum PM. Because of this, we track the minimum PM of each $\mathcal{A}$ we create.

The algorithm we use to compute the PM using the conditional DEGA has the underlying assumption that the frozen bit we want to make erroneous as a result of the error on i *does indeed* get altered to the ideal decision, i.e., we are operating under the assumption of a perfect, time and length varying, sparse matrix as our convolutional precoder. We do not use these generator matrices in our numerical experiments, and use instead a fixed time-invariant one. Empirically, we have observed that a random generator matrix performs well enough in selecting key frozen bits, and does decent in avoiding the problem caused by zero-mean information bits documented in Section III-A. However, we believe that finding these aforementioned perfect generator connection matrices may further improve the performance, but is a hard optimization problem that remains open.

When we are in a loop, meaning that we create a set $\mathcal{A}$ that we have already visited, we perturb the set with a swap finding process as follows. We look at all $j \in \mathcal{F}'$, and try to see if the PM_j with an additional, PAC-forced error on information bit i is larger than PM_i. If this is the case, we move j into $\mathcal{A}$ and i into $\mathcal{F}'$. This potentially raises the average PM, as well as enlarges our overall search space. If swaps were made in this step, and the resulting $\mathcal{A}$ is new, we go back to the first phase of the algorithm, initated with this $\mathcal{A}$. Once we are unable to discover a new $\mathcal{A}$ in any stage, we return the $\mathcal{A}^*$ with the largest minimum PM. Our simulations use the fact that we are transmitting the all zero codeword, which remains the all zero codeword even after convolution. However, the sets we construct are used for general transmissions of random messages.

A summary of the algorithm can be provided upon request. Despite the large simplifications and heuristics we implement, leveraging our principled approach to an SCL and PM based construction allows us to design extremely high performing sets. The results described in Section V support the benefits of our approach to analyze PAC codes, and we believe that with an improved approach to the optimization problem, near optimal sets could be obtained. We also believe that our design philosophy may be able to generate information sets for larger blocklengths where methods based on minimum weight words could be found lacking.

V. Numerical Results

To demonstrate the advantage of a PM based construction, we show the construction of a few sets of varying size, rate, and list size. In all our simulations, our message u is generated from a Bern($\frac{1}{2}$) source, v is created with the polynomial $P = [1, 0, 1, 1, 0, 1, 1]$, x is created via the modulation scheme $1 - 2v$ and y is the received vector under an AWGN channel. The polynomial found in the Weighted Sum (WS) [13] paper was used for the WS simulations. For these cases, our information set outperforms other, highly competitive sets. For $(128, 42)$, we obtain up to half of a dB gain against [13] for list sizes 32 and 8. The exact results are shown in Figs 2-3.

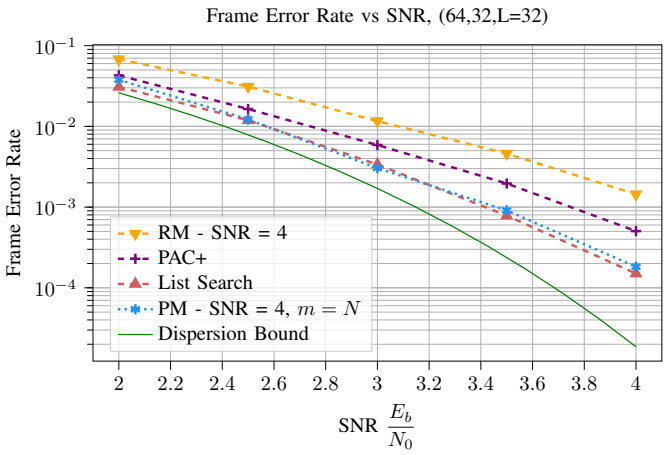


Fig. 2: PAC performance for $N = (64, 32)$ code with $L = 32$. Created with construction SNR of 4 and unbounded m. Compared with RM, PAC+, and List Search.

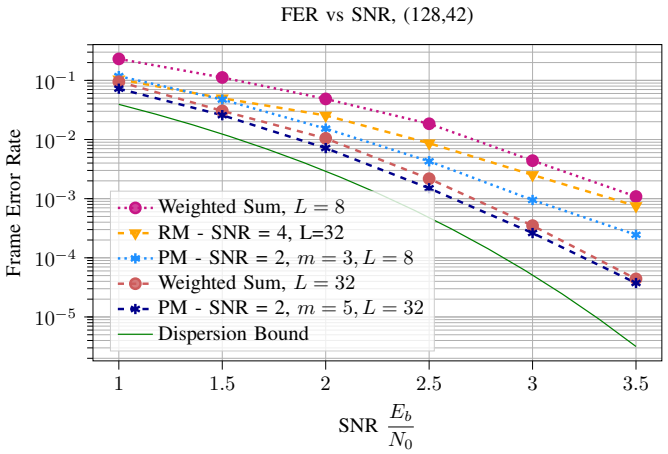


Fig. 3: PAC performance for $N = (128, 42)$ code with varying L. Created with construction SNR of 2 and $m = 5, 3$. Compared with RM and Weighted Sum.

For $(128, 42, L = 8)$, we can see that our set, which is designed based off SCL, is far more robust against a changed list size than the high performance WS set, providing value for decoders with memory or latency requirements. In fact, our set offers half of a dB in improvement over the WS with $L = 8$, and the RM set with $L = 32$. Our algorithm is very sensitive to the choice of both the design SNR and m, the parameter that limits how far out (in terms of information bits) our desired frozen bit can be, with small changes producing different sets that vary highly in performance.

The polynomial we use for our generator is common in the literature, and we believe that most random polynomials are similar and sufficient. However, exploiting gains at higher rates or blocklengths may require the creation of the matching generator. Improving the algorithm to completely address path loss and selection errors, as well as creating the matching generator, is the subject of ongoing work.

REFERENCES

[1] C. Yue, V. Miloslavskaya, M. Shirvanimoghaddam, B. Vucetic, and Y. Li, "Efficient decoders for short block length codes in 6G URLLC," arXiv preprint https://arxiv.org/pdf/2206.09572.pdf, Dec. 2022.

[2] M. C. Coskun, G. Durisi, T. Jerkovits, G. Liva, W. E. Ryan, B. Stein, and F. Steiner, "Efficient error-correcting codes in the short blocklength regime," *CoRR*, vol. abs/1812.08562, 2018. [Online]. Available: http://arxiv.org/abs/1812.08562

[3] M. C. Coskun, J. Neu, and H. D. Pfister, "Successive cancellation inactivation decoding for modified reed-muller and eBCH codes," in *Proc. of IEEE International Symposium on Information Theory*, Los Angeles, California, USA, Jun. 2020.

[4] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Transactions on Information Theory*, vol. 55, no. 7, pp. 3051–3073, 2009.

[5] E. Arıkan, "From sequential decoding to channel polarization and back again," Aug. 2019.

[6] M. Rowshan and E. Viterbo, "List viterbi decoding of PAC codes," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 3, pp. 2428–2435, Mar. 2021.

[7] M. Rowshan, A. Burg, and E. Viterbo, "Polarization-adjusted convolutional (PAC) codes: Sequential decoding vs list decoding," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 2, pp. 1434–1447, Feb. 2021.

[8] H. Yao, A. Fazeli, and A. Vardy, "List decoding of arikan's PAC codes," *CoRR*, vol. abs/2005.13711, 2020. [Online]. Available: https://arxiv.org/abs/2005.13711

[9] M. Fossorier and S. Lin, "Soft-decision decoding of linear block codes based on ordered statistics," *IEEE Transactions on Information Theory*, vol. 41, no. 5, pp. 1379–1396, 1995.

[10] C. Yue, M. Shirvanimoghaddam, B. Vucetic, and Y. Li, "Ordered-Statistics Decoding with Adaptive Gaussian Elimination Reduction for Short Codes," Dec. 2022.

[11] J. Cheng and L. Chen, "Bch based u-uv codes and its decoding," in *2021 IEEE International Symposium on Information Theory (ISIT)*, Jul. 2021, p. 1433–1438. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9517823

[12] K. R. Duffy, J. Li, and M. Medard, "Capacity-achieving guessing random additive noise decoding," *IEEE Transactions on Information Theory*, vol. 65, no. 7, pp. 4023–4040, Jul. 2019.

[13] W. Liu, L. Chen, and X. Liu, "A weighted sum based construction of PAC codes," *IEEE Communications Letters*, vol. 27, no. 1, pp. 28–31, 2023.

[14] H. Sun, E. Viterbo, and R. Liu, "Analysis of polarization-adjusted convolutional codes (PAC): A source-channel coding method," in *2021 IEEE Globecom Workshops (GC Wkshps)*, Madrid, Spain, Dec. 2021, pp. 1–6.

[15] M. Rowshan, S. H. Dau, and E. Viterbo, "On the Formation of Min-weight Codewords of Polar/PAC Codes and Its Applications," *IEEE Transactions on Information Theory*, pp. 1–1, 2023.

[16] S. Jiang, J. Wang, C. Xia, and X. Li, "Construction of PAC Codes with List-Search and Path-Splitting Critical Sets," Apr. 2023.

[17] I. Tal and A. Vardy, "How to construct polar codes," *IEEE Transactions on Information Theory*, vol. 59, no. 10, pp. 6562–6582, Oct. 2013.

[18] B. Li, H. Shen, and D. Tse, "A RM-polar codes," *CoRR*, vol. abs/1407.5483, 2014. [Online]. Available: http://arxiv.org/abs/1407.5483

[19] I. Tal and A. Vardy, "List decoding of polar codes," *IEEE Transactions on Information Theory*, vol. 61, no. 5, pp. 2213–2226, May 2015.

[20] S.-Y. Chung, T. Richardson, and R. Urbanke, "Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 657–670, Feb. 2001, conference Name: IEEE Transactions on Information Theory.

[21] R. Mori and T. Tanaka, "Performance of Polar Codes with the Construction using Density Evolution," *IEEE Communications Letters*, vol. 13, no. 7, pp. 519–521, Jul. 2009.

[22] P. Trifonov and V. Miloslavskaya, "Polar codes with dynamic frozen symbols and their decoding by directed search," in *2013 IEEE Information Theory Workshop*, Seville, Spain, Sep. 2013.

[23] P. Trifonov, "Randomized polar subcodes with optimized error coefficient," *IEEE Transactions on Communications*, vol. 68, no. 11, pp. 6714–6722, Nov. 2020.

[24] M. Rowshan, A. Burg, and E. Viterbo, "Polarization-adjusted convolutional (pac) codes: Sequential decoding vs list decoding," Feb. 2020.

[25] B. Feng, Y. Yang, J. Jiao, and Q. Zhang, "On Tail-Biting Polarization-Adjusted Convolutional (TB-PAC) Codes and Small-Sizes List Decoding," *IEEE Communications Letters*, vol. 27, no. 2, pp. 433–437, Feb. 2023.

VI. Appendix

subroutine $GetLargestPossiblePM(N, \mathcal{A}, i, SNR)$
  $CDEGA = CDEGA(N, A, error = i)$
  $j = argmax\{CDEGA : j > i, j \notin \mathcal{A})$
return $j, \mathbf{E}[PM]$
subroutine $DoAllSwaps(N, A, F', Info_{PMs}, SNR)$
  for $i \in A$ **do**
    Find $argmax_j(PMs[j] : j \in \mathcal{F}'$, u_i uses $v_j)$
    if $PMs[j] > Info_{PMs}[i]$ **then**
      $\mathcal{A} = \mathcal{A} \cup j$
      $\mathcal{F}' = \mathcal{F}'/j$
      $\mathcal{A} = \mathcal{A}/i$
      $\mathcal{F}' = \mathcal{F}' \cup i$
    end
  end
return $j, \mathbf{E}[PM]$

TABLE I: Frozen Sets (Hexadecimal)

(64,32,SNR=4,m=-1)	5137F0777177F
(128,42,SNR=2,m=5)	10003071F0007071F033F1F7F
(128,42,SNR=2,m=3)	10001071700070717071F7FFF

Algorithm 1: PM Based Information Set Constructor

input : N Code Length, K Information Bits, SNR Design Length, ϵ Search Bits
output: Info Set $\mathcal{A}$

$\mathcal{A}_{All} = []$
$PMs_{All} = []$
$A_{start} = DEGA(N, K)$
$\mathcal{A} = A_{start}$
$\mathcal{F}'$ = NextϵLargestDEGAValues(N,K)
while *True* **do**
  $Info_{PMs} = []$
  for $i \in \mathcal{A}$ **do**
    $PM_i = GetLargestPossiblePM(N, \mathcal{A}, i, SNR)$
    $append(Info_{PMs}, (PM_i, i))$
  end
  $WorstInfoBit = argmin(Info_{PMs})$// The bit that causes lowest PM
  $MinPM = min(Info_{PMs})$ // The Lowest PM
  // If this A has already been found before, i.e. a cycle is created, we need to reroute. If it's a new A then we can ignore
  if $\mathcal{A} \in \mathcal{A}_{All}$ **then**
    $\mathcal{A}* = argmax_{PMs_{All}}(\mathcal{A}_{All})$// Set A* as the Info Set that had the largest minimum metric
    $\mathcal{A} = DoAllSwaps(N, \mathcal{A}*, Info_{PMs}, SNR)$
    if $\mathcal{A} \in \mathcal{A}_{All}$ **then**
      $Break$
    end
    // If even this A has already been found before, i.e. a cycle would still be created, end, otherwise you restart but with this (A w/ swaps) as the A to look at
    $append(\mathcal{A}_{All}, \mathcal{A})$
    $append(PMs_{All}, MinPM)$
  else
    $append(\mathcal{A}_{All}, \mathcal{A})$
    $append(PMs_{All}, MinPM)$
    $\mathcal{A} = \mathcal{A}/WorstInfoBit$// The Info Bit becomes frozen and vice versa
    $\mathcal{F}' = \mathcal{F}' \cup WorstInfoBit$
    $Froz_{PMs} = []$
    for $j \in \mathcal{F}'$ **do**
      $PM_j = GetLargestPossiblePM(N, \mathcal{F}, j, SNR)$
      $append(Froz_{PMs}, (PM_j, j))$
    end
    $BestFrozBit = argmax(Froz_{PMs})$
    $\mathcal{A} = \mathcal{A} \cup BestFrozBit$
    $\mathcal{F}' = \mathcal{F}'/BestFrozBit$
  end
end
return $argmax(\mathcal{A}_{All})$